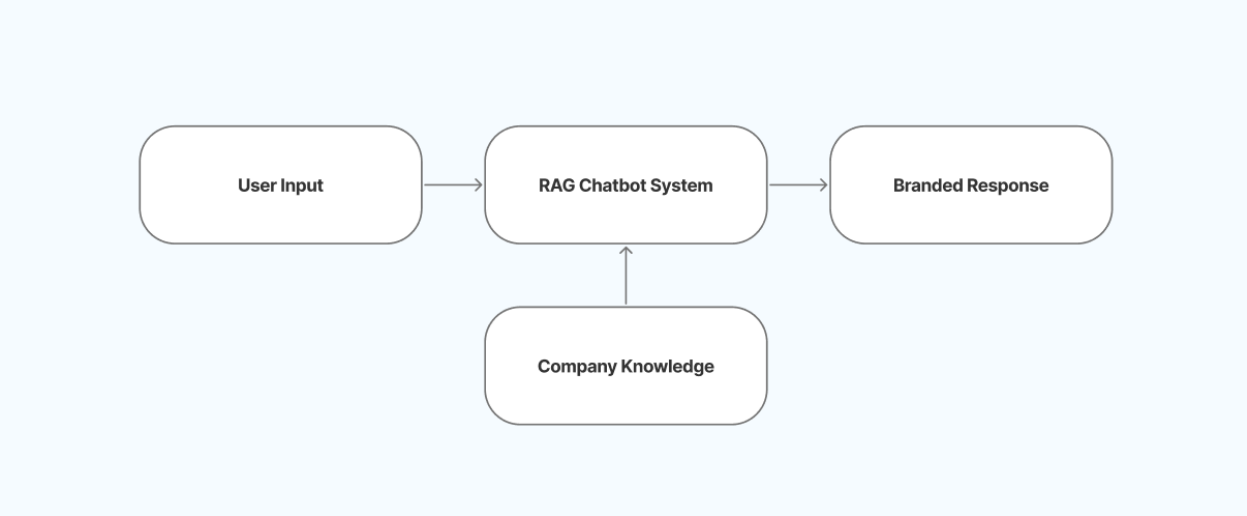
1. Introduction

This project explores the application of retrieval-augmented generation (RAG) techniques to build an intelligent Q&A chatbot using company-specific knowledge. As part of the R&D rad-gp-c25-p-i6 project, the objective was to design a chatbot capable of delivering contextual, consistent, and accurate responses by integrating prompt templating, conversational memory, embeddings, and vector databases. The result is a functioning prototype that demonstrates the practical potential of combining these technologies in real-world support scenarios.



---

2. Applied Techniques

2.1 Prompt Design and Templating

To ensure brand consistency and user clarity, The project implemented structured system prompts that define tone, format, and fallback behavior. These templates allowed the chatbot to produce standardized and professional responses while staying aligned with define communication style.

2.2 Buffer Memory

Conversational memory was introduced using buffer memory via LangChain's message history. This enabled the chatbot to retain context across multiple user turns and deliver more coherent, personalized responses.

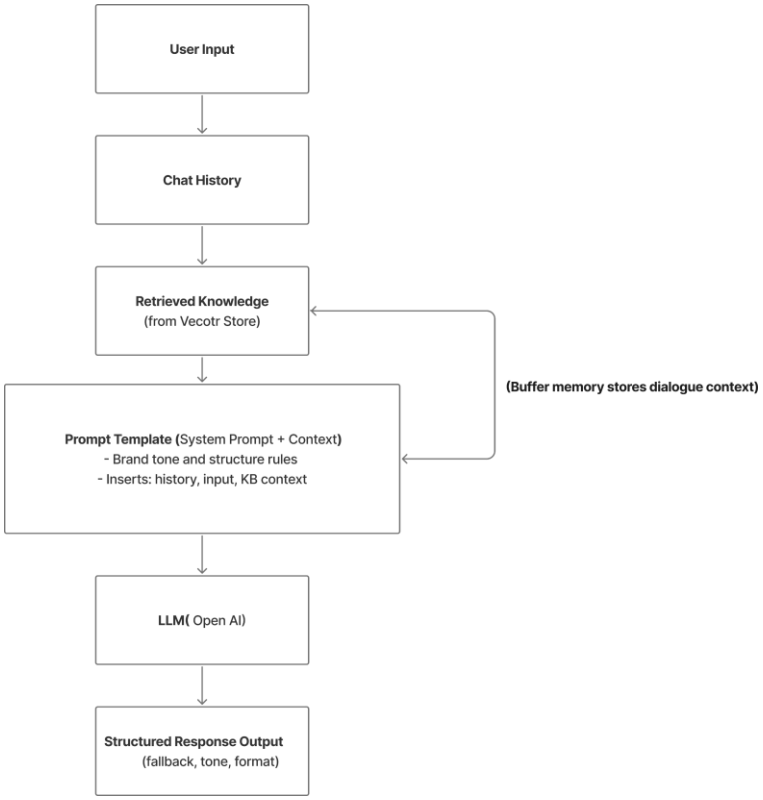
2.3 Embeddings and Vector Database

The chatbot uses a text file containing company information as its primary knowledge base. This content is processed using a transformer-based embedding model to convert text into high-dimensional vectors. These vectors are stored in a FAISS vector database, enabling fast and accurate semantic search. When a user submits a query, the system retrieves the most relevant content from this database, ensuring responses are enriched with trusted, internal company knowledge.

---

3. Chatbot Workflow Summary

1. **User Input:** The user enters a message through the terminal interface.
2. **Session History:** Chat history is retrieved or created using the session ID.
3. **Knowledge Retrieval:** Relevant documents are fetched from the FAISS index based on the input.
4. **Prompt Construction:** A prompt is built using system rules, chat history, and the user's input.
5. **LLM Response:** The prompt is sent to GPT-3.5-Turbo to generate a response.
6. **Fallback Detection:** If the response contains fallback phrases, a live agent handoff message is shown.



---

4. Key Outcomes

- Prompt templating enforced clarity and tone control.
- Memory enabled dynamic multi-turn interactions.
- Embeddings improved response relevance by grounding answers in trusted documentation.
- Fallback logic ensured reliability by gracefully handling edge cases.

---

5. Conclusion

This project demonstrates how RAG-based architectures combined with memory and templating can be used to build a production ready chatbot capable of handling complex, domain-specific queries.